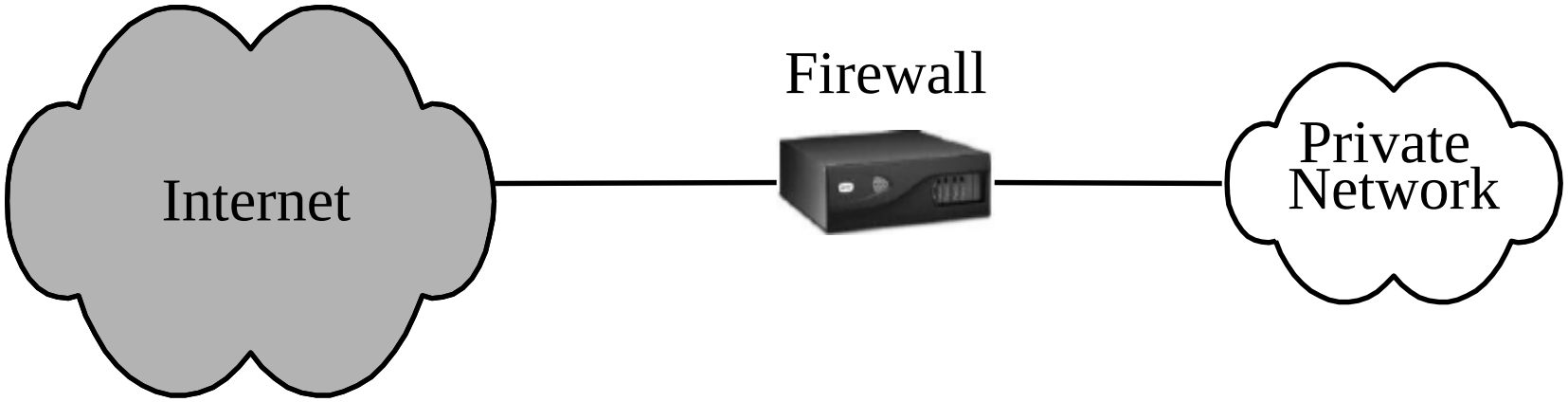
Firewall Design Methods

Haipeng Dai

haipengdai@nju.edu.cn
313 CS Building
Department of Computer Science and Technology
Nanjing University

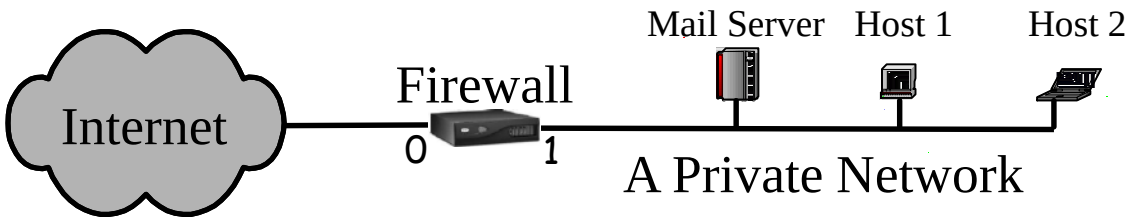
Security Guard for Private Buildings

Security Guard for Private Networks



- **Location**: connects Internet and private network
- **Function**: maps every packet to a decision - accept or discard
- **Configuration**: a sequence of rules written by administrator

Firewall Example



Interface	Source IP	Dest. IP	Dest. Port	Protocol	Decision
0	any	mail server	25	TCP	accept
0	malicious hosts	any	any	any	discard
1	{host1, host2}	any	80	TCP	accept
any	any	any	any	any	accept

- Rules are conflicting
- First match: decision for packet = decision of first matching rule
- Order matters

Real-life Firewalls are Complex

```
523: conduit permit tcp host 100.77.28.87 eq 8100 any
524: conduit permit tcp host 100.77.28.87 eq 8110 any
525: conduit permit tcp host 100.77.28.84 eq ftp host 207.115.175.244
526: conduit permit tcp host 100.77.28.84 eq telnet host 198.215.163.20
527: conduit permit tcp host 100.77.28.84 eq ftp host 198.215.163.20
528: conduit permit tcp host 100.77.28.84 eq telnet host 198.215.163.21
529: conduit permit tcp host 100.77.28.84 eq ftp host 198.215.163.21
530: conduit permit tcp host 100.77.28.87 eq www host 207.115.175.244
531: conduit permit tcp host 100.77.28.87 eq telnet host 207.115.175.244
532: conduit permit tcp host 100.77.28.87 eq 443 host 207.115.175.244
533: conduit permit tcp host 100.77.28.87 eq ftp host 207.115.175.244
534: conduit permit tcp host 100.77.28.87 eq www host 205.170.235.0
535: conduit permit tcp host 100.77.28.87 eq 443 host 205.170.235.0
536: conduit permit tcp host 100.77.28.87 eq ftp host 198.215.163.20
537: conduit permit tcp host 100.77.28.87 eq ftp host 198.215.163.21
538: conduit permit tcp host 100.77.28.88 eq telnet 12.20.51.0 255.255.255.0
539: conduit permit tcp host 100.77.28.88 eq ftp 12.20.51.0 255.255.255.0
540: conduit permit tcp host 100.77.28.88 eq www 12.20.51.0 255.255.255.0
541: conduit permit tcp host 100.77.28.88 eq 13292 12.20.51.0 255.255.255.0
542: conduit permit tcp host 100.77.28.88 eq 443 12.20.51.0 255.255.255.0
543: conduit permit tcp host 100.77.28.84 eq telnet 12.20.51.0 255.255.255.0
544: conduit permit tcp host 100.77.28.84 eq ftp 12.20.51.0 255.255.255.0
545: conduit permit tcp host 100.77.28.85 eq www 12.20.51.0 255.255.255.0
546: conduit permit tcp host 100.77.28.85 eq telnet 12.20.51.0 255.255.255.0
547: conduit permit tcp host 100.77.28.85 eq 443 12.20.51.0 255.255.255.0
548: conduit permit tcp host 100.77.28.85 eq ftp 12.20.51.0 255.255.255.0
549: conduit permit tcp host 100.77.28.87 eq www 12.20.51.0 255.255.255.0
550: conduit permit tcp host 100.77.28.87 eq telnet 12.20.51.0 255.255.255.0
551: conduit permit tcp host 100.77.28.87 eq 443 12.20.51.0 255.255.255.0
```

Number of rules can be large

Legacy rules

Cascade impact of change

Problem

- As a result, firewall rules are hard to specify correctly
  hard to understand correctly
  hard to change correctly

- Consequently, firewall configuration errors are common
  - Most firewalls are poorly designed with errors [Wool'04]

- Firewall errors are unacceptable
  - Accept malicious packets: lose security
  - Discard legitimate packets: disrupt business

- Problem: How to design firewalls?

State-of-the-art

- Industry: tweak and pray

- Academia: analyze rules
  - Such as conflict detection ([HSP 00] [EM 01] [BV 02])
    anomaly detection ([AH 03] [AH 04])

Structured Firewall Design: Motivation

- The convention of designing a firewall directly as a sequence of conflicting rules has been taken for granted
- We point out that this convention is BAD.
- Why: this convention has three major issues
  - Consistency issue
  - Completeness issue
  - Compactness issue

Consistency Issue

Interface	Source IP	Dest. IP	Dest. Port	Protocol	Decision
0	**any**	**mail server**	**25**	**TCP**	**accept**
0	**malicious hosts**	**any**	**any**	**any**	**discard**
1	{host1, host2}	any	80	TCP	accept
any	any	any	any	any	accept

- This firewall accepts email from malicious hosts!
- This is wrong (assuming this firewall is required to discard all packets from malicious hosts)

Consistency Issue

Interface	Source IP	Dest. IP	Dest. Port	Protocol	Decision
0	**any**	**mail server**	**25**	**TCP**	**accept**
0	**malicious hosts**	**any**	**any**	**any**	**discard**
1	{host1, host2}	any	80	TCP	accept
any	any	any	any	any	accept

- This firewall accepts email from malicious hosts!
- This is wrong (assuming this firewall is required to discard all packets from malicious hosts)
- We should swap the first two rules
- Consistency issue: hard to ensure rules are ordered correctly

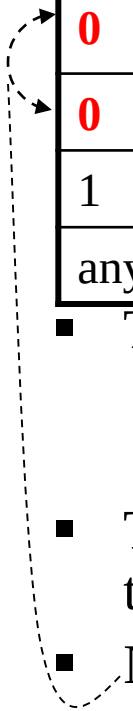
Completeness Issue

Interface	Source IP	Dest. IP	Dest. Port	Protocol	Decision
0	malicious hosts	any	any	any	discard
0	**any**	**mail server**	**25**	**TCP**	**accept**
1	{host1, host2}	any	80	TCP	accept
any	**any**	**any**	**any**	**any**	**accept**

- This firewall accepts
  - non-email packets to the email server!
  - email packets to hosts other than the email server!
- This is wrong (assuming this firewall is required to discard the above two types of packets)

Completeness Issue

Interface	Source IP	Dest. IP	Dest. Port	Protocol	Decision
0	malicious hosts	any	any	any	discard
0	any	mail server	25	TCP	accept
0	**any**	**mail server**	**any**	**any**	**discard**
0	**any**	**any**	**25**	**TCP**	**discard**
1	{host1, host2}	any	80	TCP	accept
any	any	any	any	any	accept



- This firewall accepts
  - non-email packets to the email server!
  - email packets to hosts other than the email server!
- This is wrong (assuming this firewall is required to discard the above two types of packets)
- Need to add two more rules
- Completeness issue: hard to ensure all necessary rules are included

Compactness Issue

Interface	Source IP	Dest. IP	Dest. Port	Protocol	Decision
0	malicious hosts	any	any	any	discard
0	any	mail server	25	TCP	accept
0	any	mail server	any	any	discard
0	any	any	25	TCP	discard
1	**{host1, host2}**	**any**	**80**	**TCP**	**accept**
any	any	any	any	any	accept

- This rule is redundant!

- Compactness issue: hard to ensure all rules are needed

Consistency, Completeness, and Compactness

- Consistency and completeness issues cause firewall errors
- Compactness issue causes low firewall performance

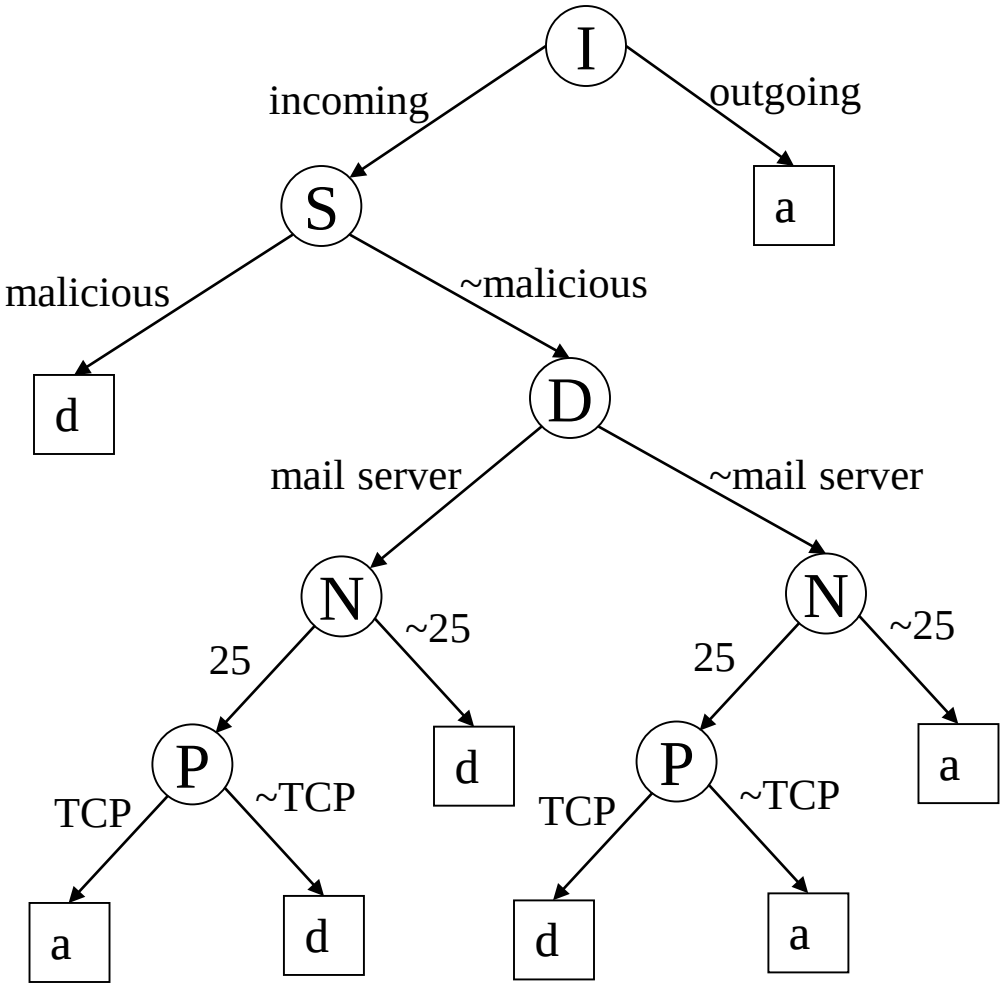
  - Less rules, faster decision
  - Fast firewalls use TCAM (Ternary Content Addressable Memory)
- Solution: Structured Firewall Design

Structured Firewall Design

Step 1: Formally specify the function of a firewall using a Firewall Decision Diagram (FDD)

Step 2: Use a series of 3 algorithms to automatically convert the FDD to a compact sequence of rules

Firewall Decision Diagram (FDD)



I: Interface

S: Source IP address

D: Dest. IP address

N: Dest. port number

P: Protocol type

a: accept

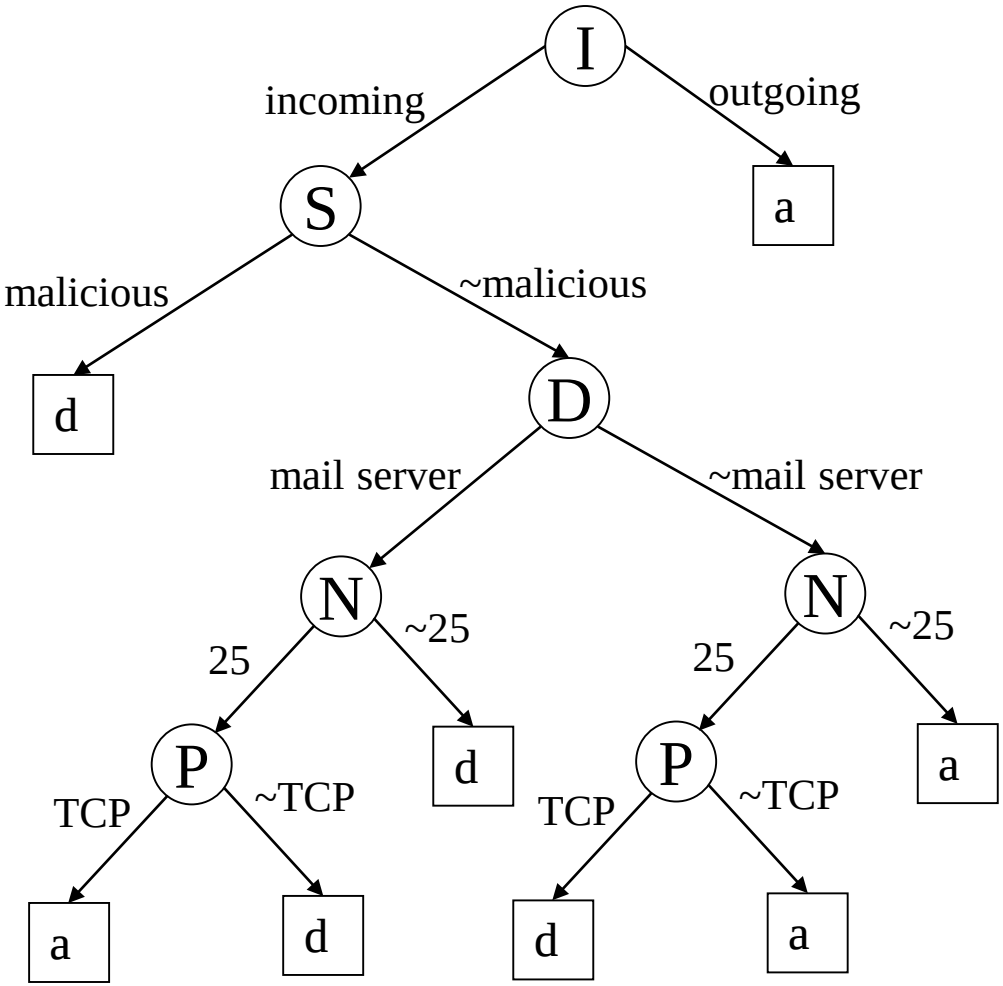
d: discard

Two important properties:

1. Consistency Property: addresses the consistency issue

2. Completeness Property: addresses the completeness issue

FDD vs. A Sequence of Conflicting Rules



I	Source IP	Dest. IP	Dest. Port	Protocol	Decision
0	malicious hosts	any	any	any	d
0	any	mail server	25	TCP	a
0	any	mail server	any	any	d
0	any	any	25	TCP	d
any	any	any	any	any	a

FDD: easy to understand

easy to update

"Goto Statement Considered Harmful"

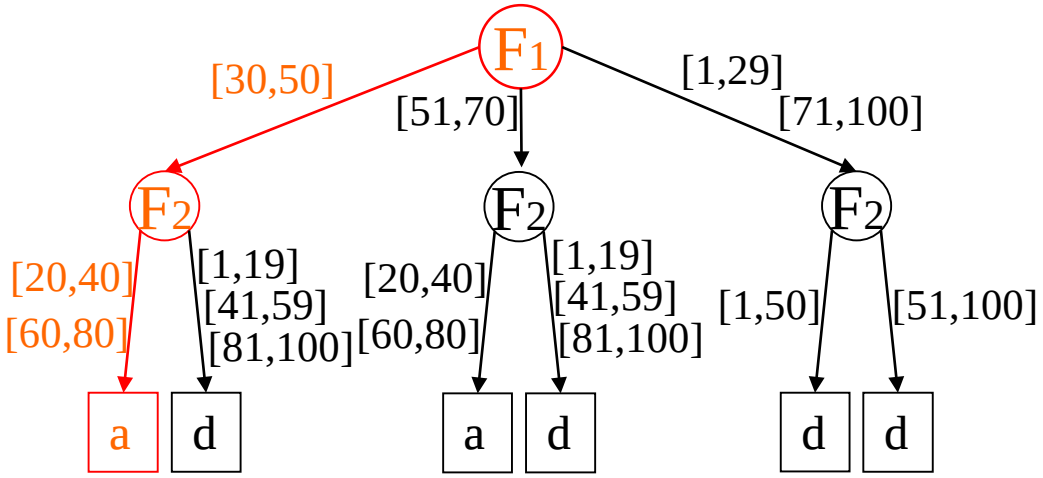
Edsger W. Dijkstra (1968)

Compatible with Existing Firewalls

- Current firewall hardware and software takes a sequence of rules

- We can convert an FDD to a sequence of rules

FDD and Rules



F_1, F_2 : packet fields

F_1's domain
=F_2's domain
=[1,100]

$F_1 \in [30,50] \wedge F_2 \in [20,40] \rightarrow a$

$F_1 \in [30,50] \wedge F_2 \in [60,80] \rightarrow a$

…

Total: 14 simple rules

General rule format:

$$F_1 \in S_1 \wedge \ldots \wedge F_d \in S_d \rightarrow a/d$$

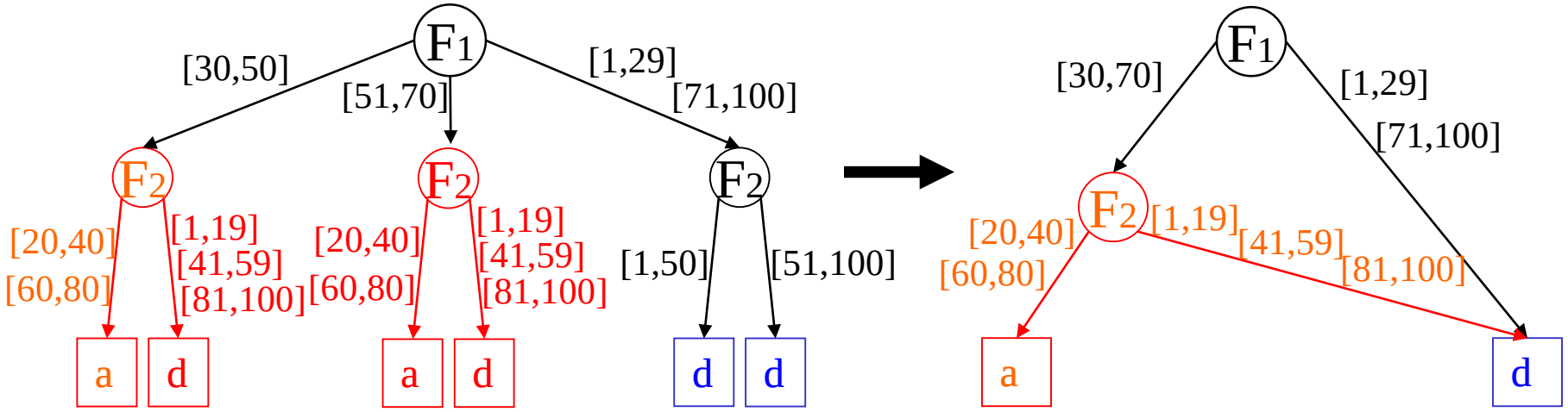
Simple rule: each S_i is one interval

Firewall implementations requires simple rules.

Reduce Number of Rules

- Three techniques:
  - FDD reduction
  - FDD marking
  - Redundancy removal

Optimization Ⅰ: FDD Reduction



- 14 simple rules ⟶ 7 simple rules
- Similar to BDD (Binary Decision Diagram) reduction [Bryant 1986]

Optimization Ⅱ: FDD Marking

- For each non-terminal node, mark one of its outgoing edges “ALL”.

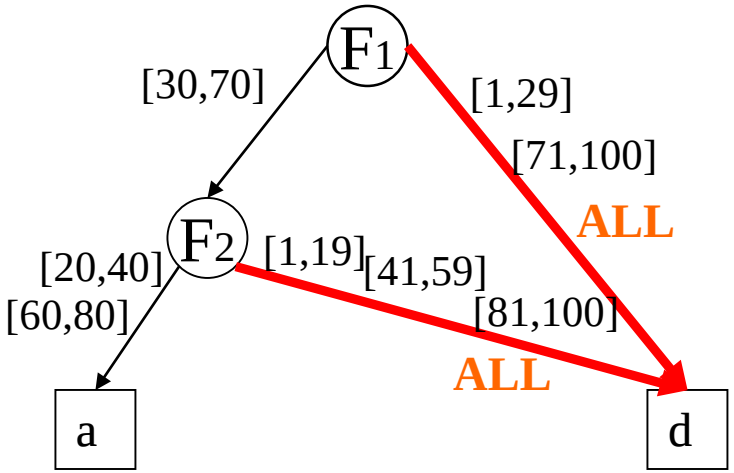
- In depth-first traversal, marked edges are traversed last:

$F_1 \in [30,\ 70] \wedge F_2 \in [20,\ 40] \rightarrow a$

$F_1 \in [30,\ 70] \wedge F_2 \in [60,\ 80] \rightarrow a$

$F_1 \in [30,\ 70] \wedge F_2 \in ALL \rightarrow d$

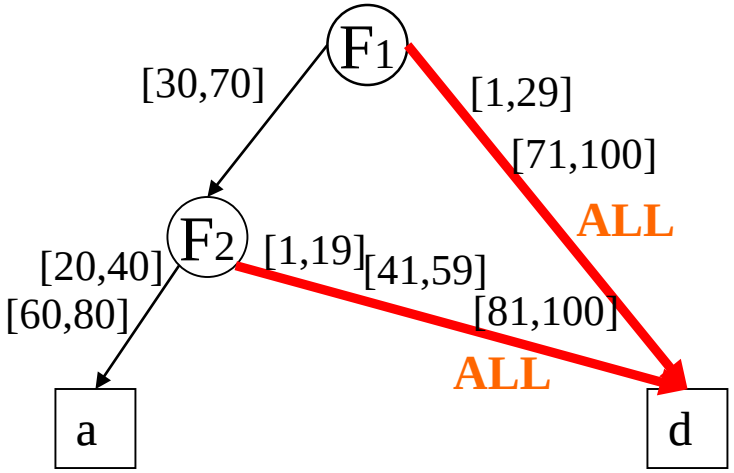
$F_1 \in ALL \wedge F_2 \in [1,\ 100] \rightarrow d$



- 7 simple rules ⟶ 4 simple rules

- We have an optimal marking algorithm (complexity: O(V+E))

Optimization III: Redundancy Removal



$F_1 \in [30,\ 70] \wedge F_2 \in [20,\ 40] \rightarrow a$

$F_1 \in [30,\ 70] \wedge F_2 \in [60,\ 80] \rightarrow a$

$F_1 \in [30,\ 70] \wedge F_2 \in ALL \rightarrow d$

$F_1 \in ALL \wedge F_2 \in [1,\ 100] \rightarrow d$

This rule is redundant!

- 4 simple rules ⟶ 3 simple rules
- We have an algorithm that can remove all redundant rules

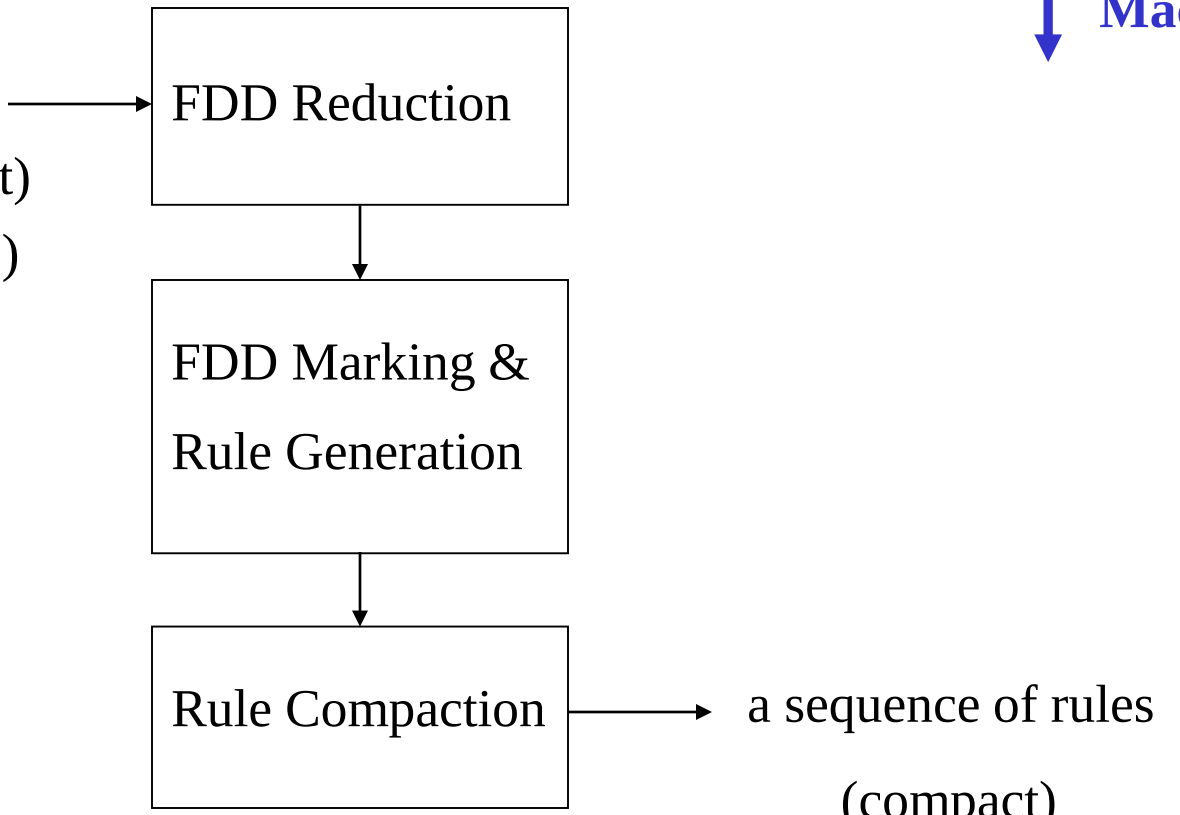
Summary of Structured Firewall Design

Step 1: Formally specify the function of a firewall using an FDD

Human

Machine

Step 2: FDD (consistent) (complete)



Not Just Firewalls……

- Routers have packet classifiers too.
  - Access control
  - Accounting
  - Quality of Service

Diverse Firewall Design

Diverse Firewall Design

- Two steps:
  - Step 1: give same requirement to multiple teams to design firewalls
  - Step 2: compare multiple firewalls to discover all functional discrepancies
- Inspired by N-version programming [Avizienis'77]
- Only deploy one firewall because we can discover all discrepancies
- Technical Challenge:

  How to discover all the discrepancies between two given firewalls?

Example

- Firewall A:

$$F_1 \in [1, \ 50] \wedge F_2 \in [1, \ 60] \rightarrow a$$
$$F_1 \in [1, 100] \wedge F_2 \in [1, 100] \rightarrow d$$

- Firewall B:

$$F_1 \in [1, \ 30] \wedge F_2 \in [1, \ 20] \rightarrow a$$
$$F_1 \in [1, \ 30] \wedge F_2 \in [1, 100] \rightarrow d$$
$$F_1 \in [1, 100] \wedge F_2 \in [1, \ 40] \rightarrow a$$
$$F_1 \in [1, 100] \wedge F_2 \in [1, 100] \rightarrow d$$

- Discrepancies between A and B:

$$F_1 \in [1, \ 30] \wedge F_2 \in [21, 60] \rightarrow a/d$$
$$F_1 \in [31, \ 50] \wedge F_2 \in [41, 60] \rightarrow a/d$$
$$F_1 \in [51,100] \wedge F_2 \in [1, \ 40] \rightarrow d/a$$

Comparing Two Firewalls

- Step 1: FDD construction

  construct an equivalent FDD from each firewall

- Step 2: FDD shaping

  make the two FDDs semi-isomorphic

- Step 3: FDD comparison

  compare the two semi-isomorphic FDDs for discrepancies

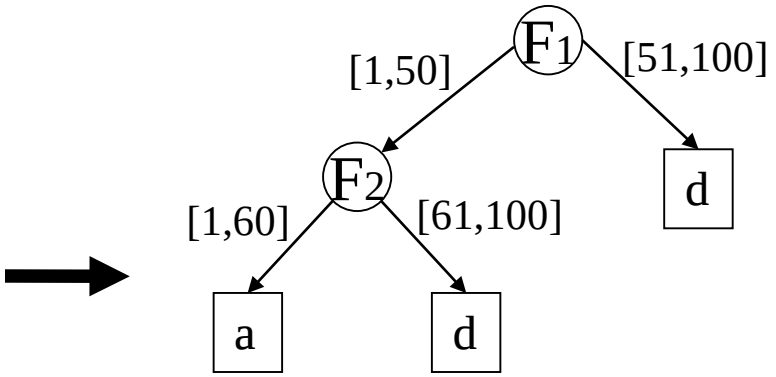
Step 1: FDD Construction

- FDD Construction Algorithm
  - Input: a firewall of a sequence of rules
  - Output: an equivalent FDD

Firewall A:

$F_1 \in [1, 50] \wedge F_2 \in [1, 60] \rightarrow a$

$F_1 \in [1, 100] \wedge F_2 \in [1, 100] \rightarrow d$



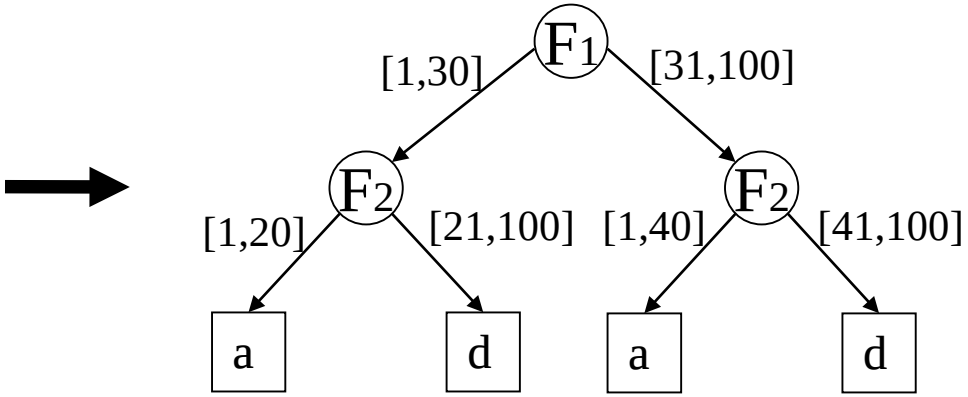
Firewall B:

$F_1 \in [1, 30] \wedge F_2 \in [1, 20] \rightarrow a$

$F_1 \in [1, 30] \wedge F_2 \in [1, 100] \rightarrow d$

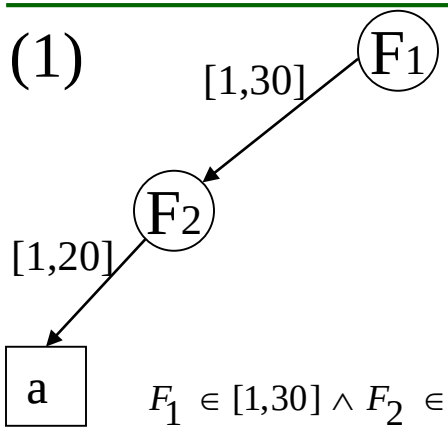
$F_1 \in [1, 100] \wedge F_2 \in [1, 40] \rightarrow a$

$F_1 \in [1, 100] \wedge F_2 \in [1, 100] \rightarrow d$



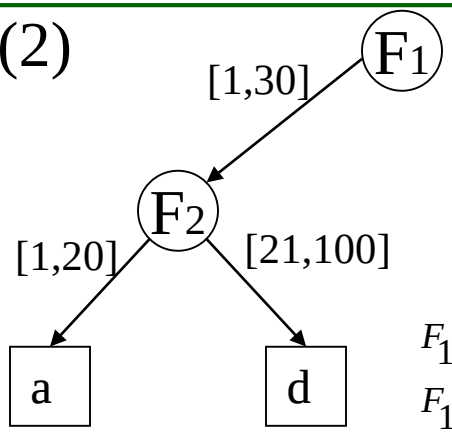
Constructing FDD

(1)



$F_1 \in [1,30] \wedge F_2 \in [1,20] \rightarrow a$

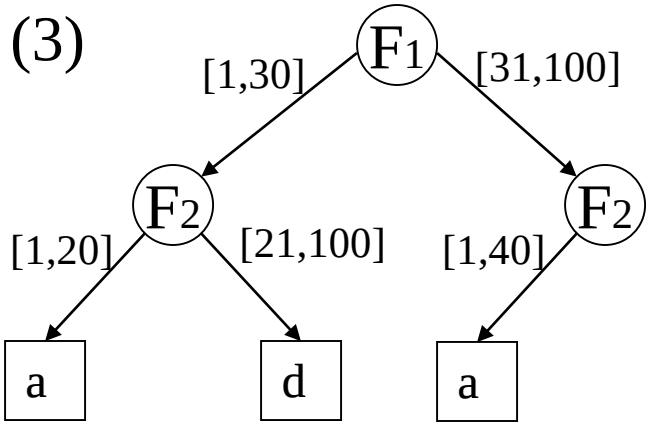
(2)



$F_1 \in [1,30] \wedge F_2 \in [1,20] \rightarrow a$

$F_1 \in [1,30] \wedge F_2 \in [1,100] \rightarrow d$

(3)

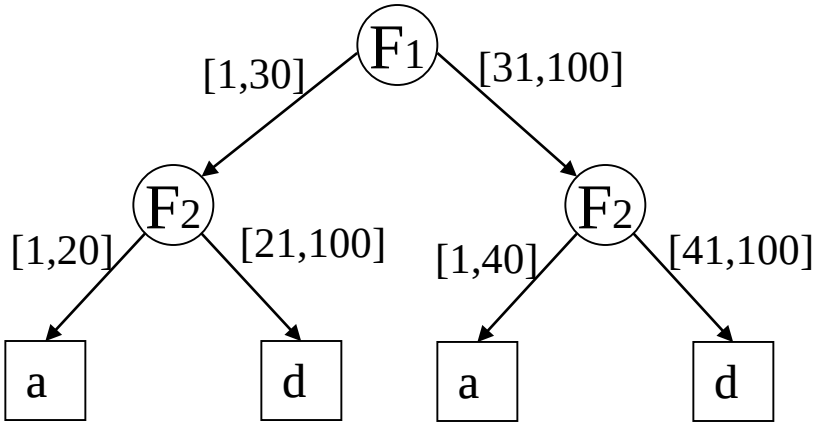


$F_1 \in [1,30] \wedge F_2 \in [1,20] \rightarrow a$

$F_1 \in [1,30] \wedge F_2 \in [1,100] \rightarrow d$

$F_1 \in [1,100] \wedge F_2 \in [1,40] \rightarrow a$

(4)



$F_1 \in [1,30] \wedge F_2 \in [1,20] \rightarrow a$

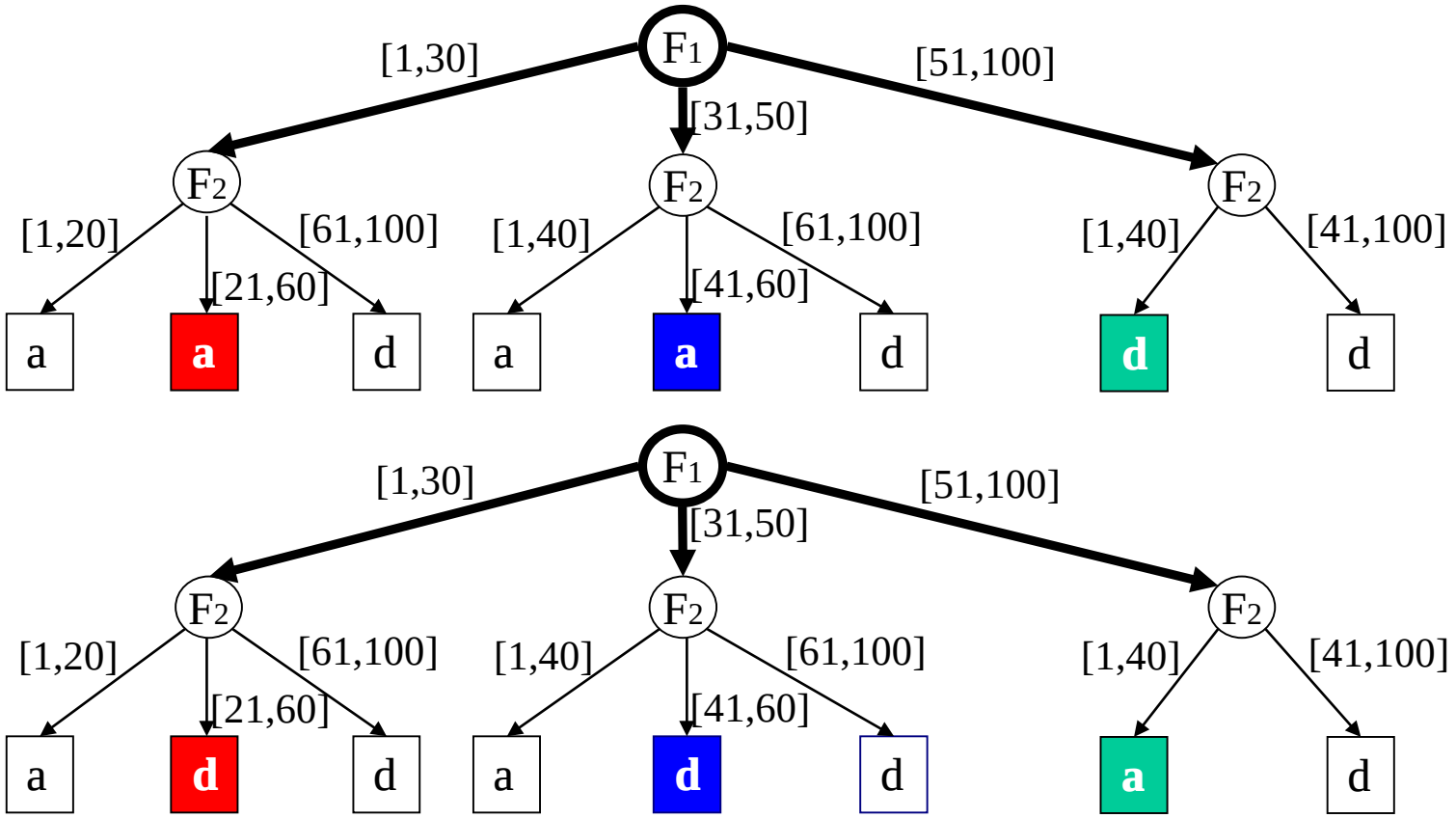
$F_1 \in [1,30] \wedge F_2 \in [1,100] \rightarrow d$

$F_1 \in [1,100] \wedge F_2 \in [1,40] \rightarrow a$

$F_1 \in [1,100] \wedge F_2 \in [1,100] \rightarrow d$

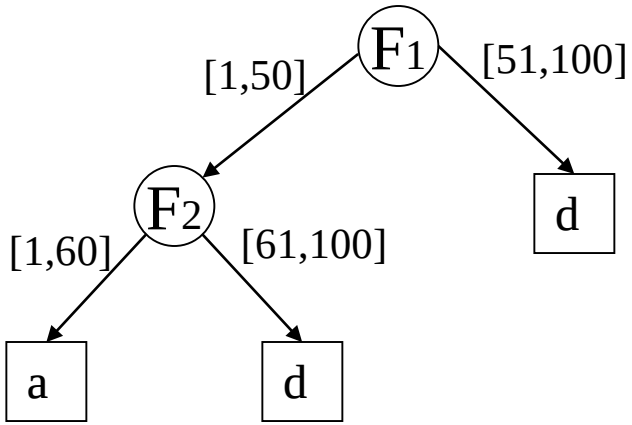
Step 2: FDD Shaping

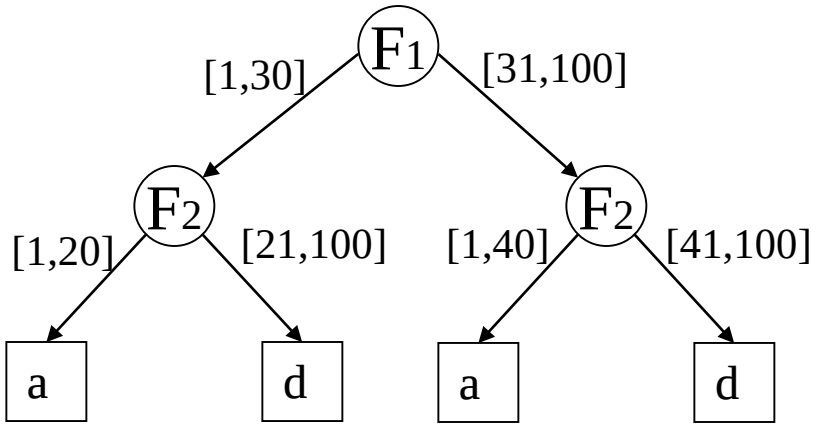
- Make two FDDs semi-isomorphic
- Semi-isomorphic FDDs: exactly same except labels of terminal nodes



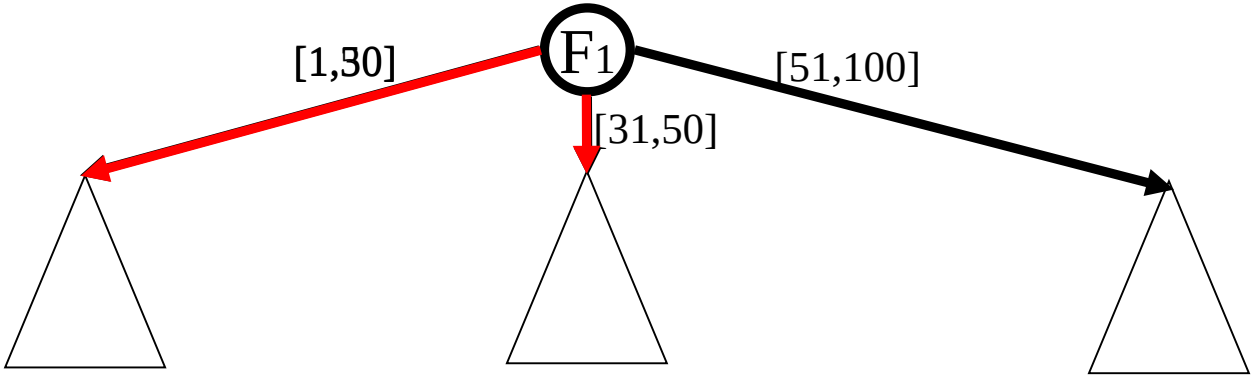
FDD Shaping

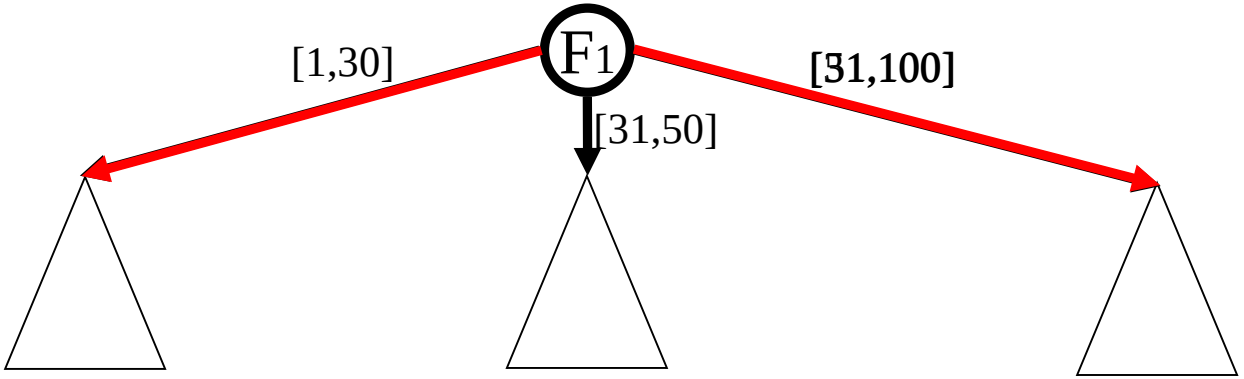
- Example: make these FDDs semi-isomorphic



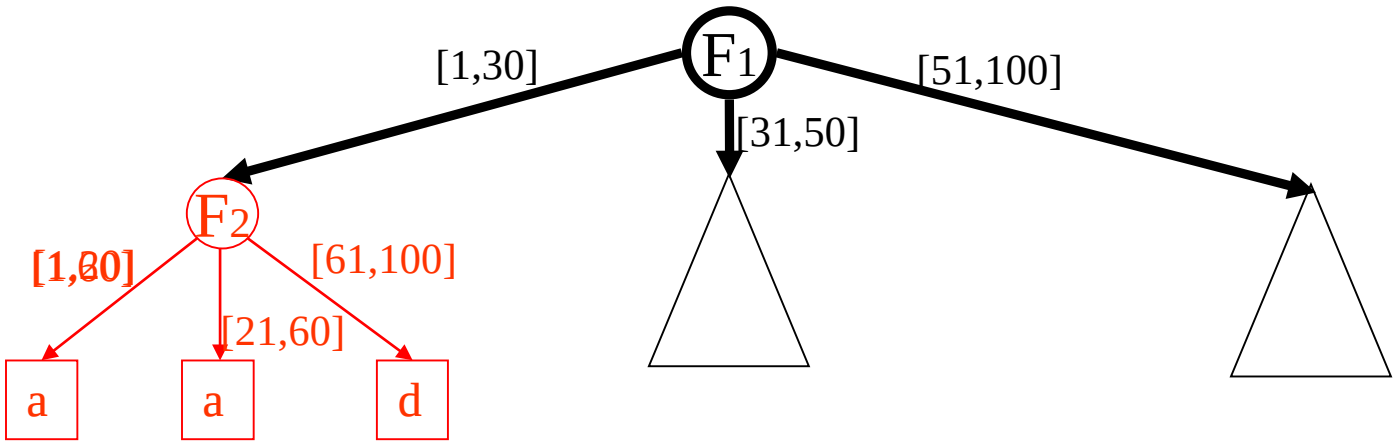


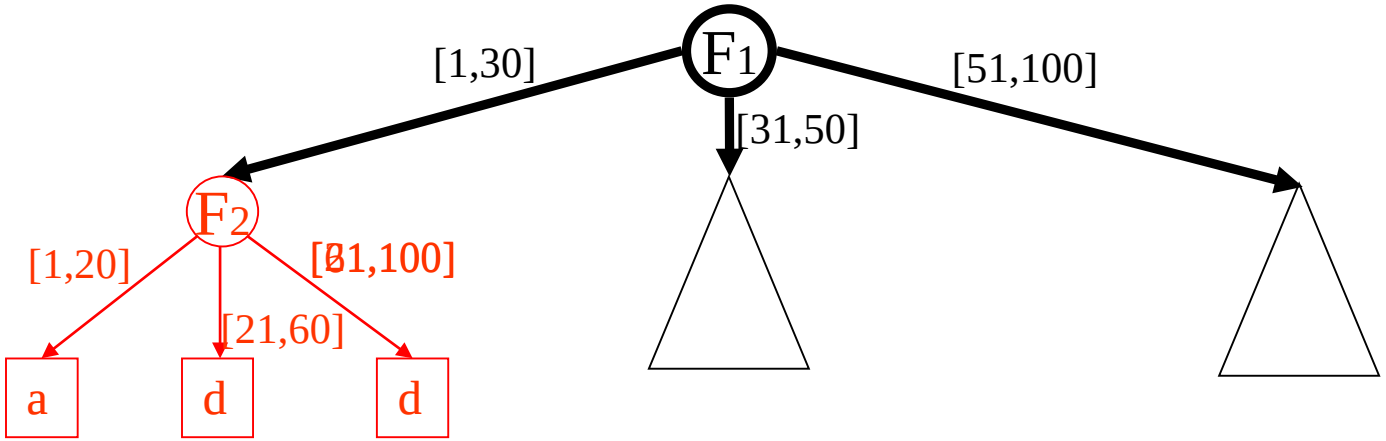
FDD Shaping



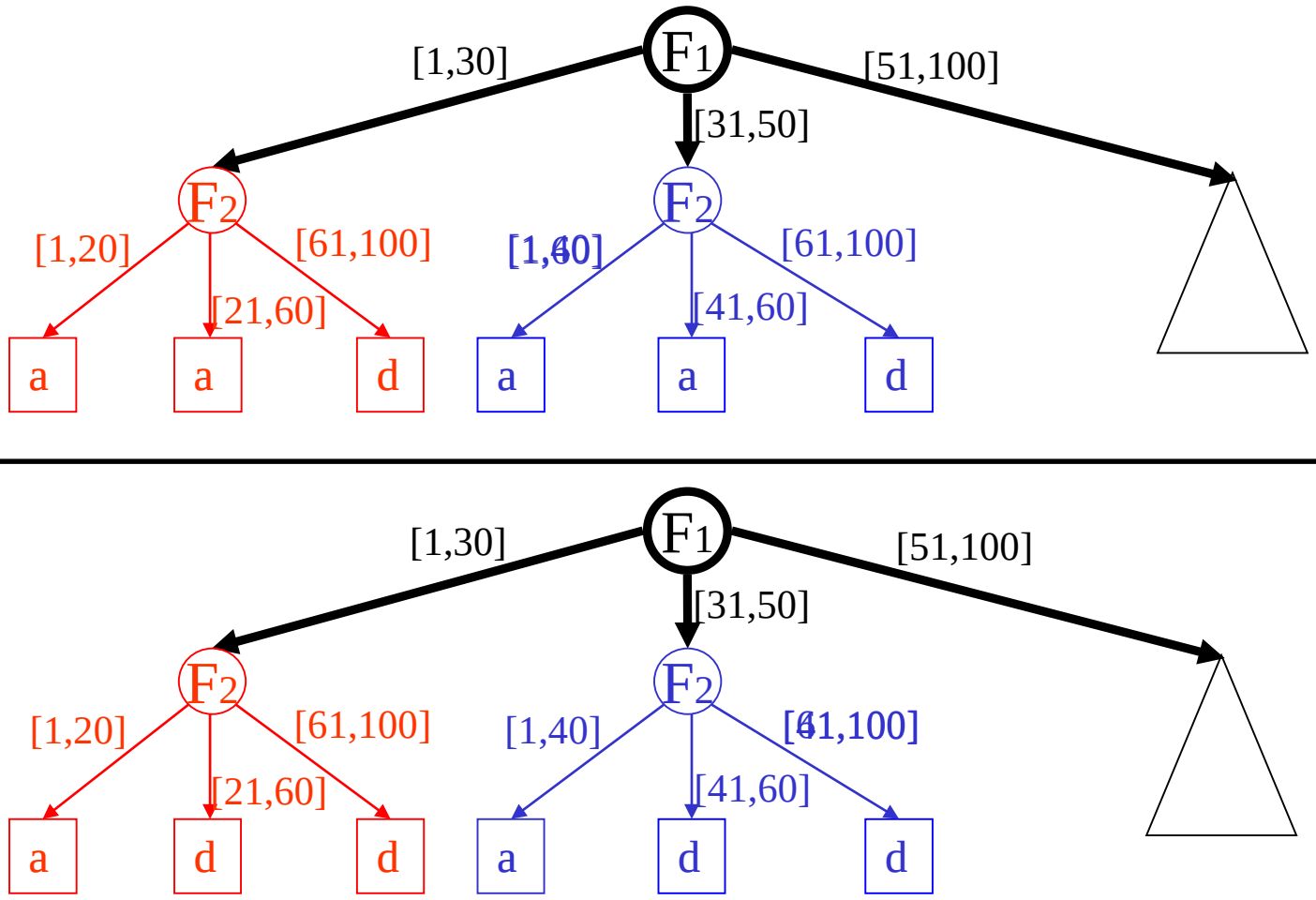


FDD Shaping

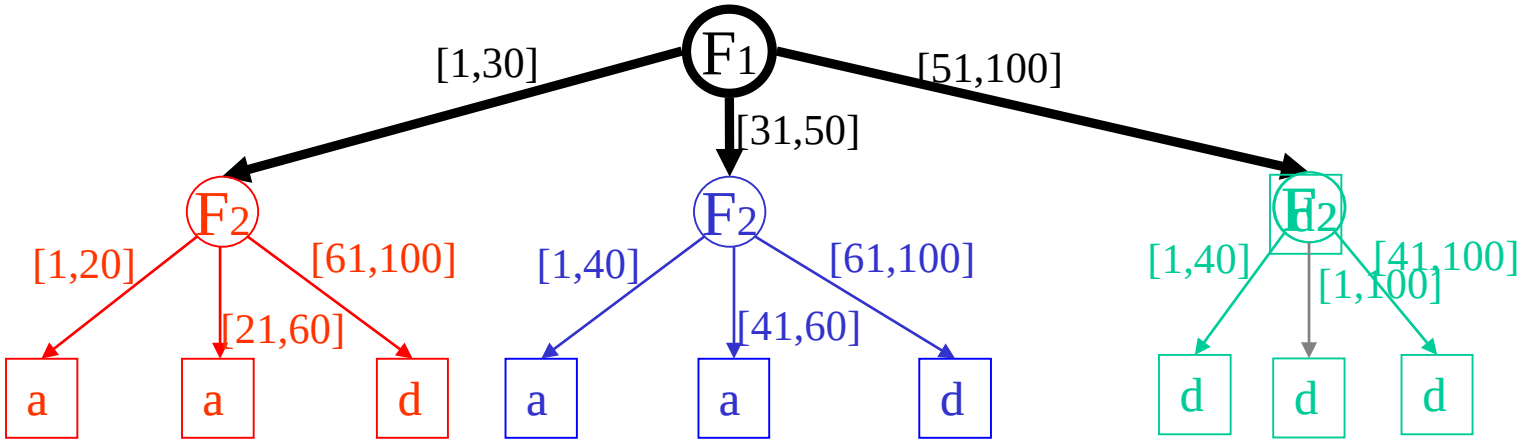

F1
[1,30]
[31,50]
[51,100]
F2
[1,20]
[21,60]
[61,100]
a
a
d

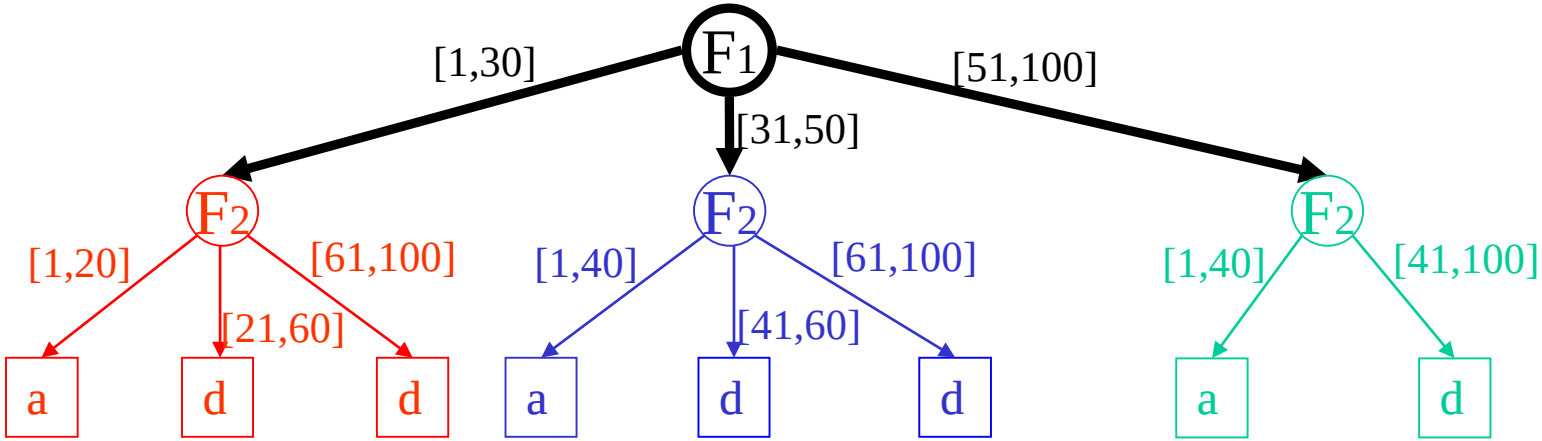

F1
[1,30]
[31,50]
[51,100]
F2
[1,20]
[21,60]
[61,100]
a
d
d

FDD Shaping


F1
[1,30]
[31,50]
[51,100]
F2
[1,20]
[21,60]
[61,100]
a
a
d
F2
[1,40]
[41,60]
[61,100]
a
a
d
F1
[1,30]
[31,50]
[51,100]
F2
[1,20]
[21,60]
[61,100]
a
d
d
F2
[1,40]
[41,60]
[61,100]
a
d
d

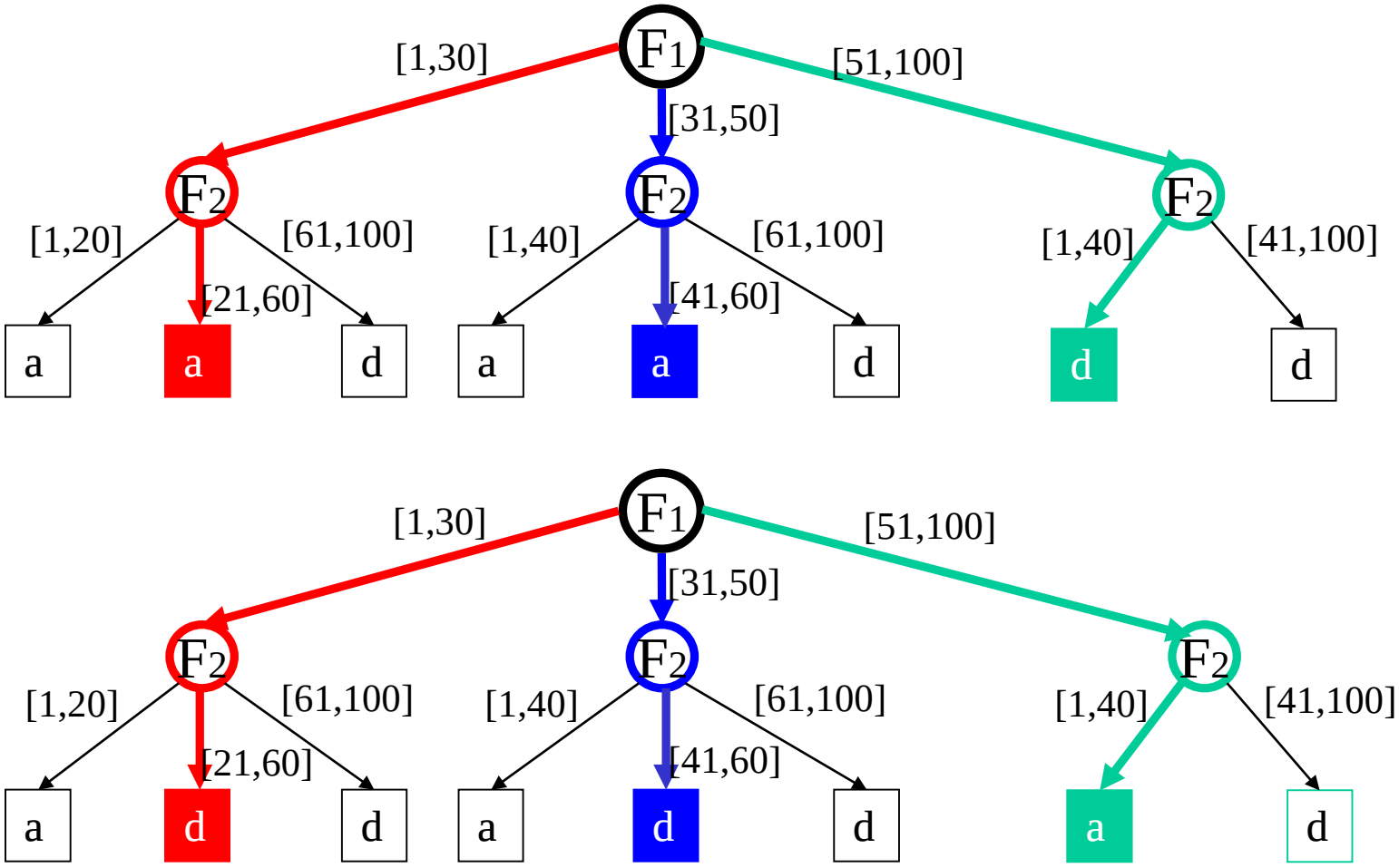
FDD Shaping


F1
[1,30]
[31,50]
[51,100]
F2
[1,20]
[21,60]
[61,100]
[1,40]
[41,60]
[61,100]
[1,40]
[1,100]
[41,100]
a
a
d
a
a
d
d
d
d


F1
[1,30]
[31,50]
[51,100]
F2
[1,20]
[21,60]
[61,100]
[1,40]
[41,60]
[61,100]
[1,40]
[41,100]
a
d
d
a
d
d
a
d

Step 3: FDD Comparison

Compare two semi-isomorphic FDDs for discrepancies



Complexity Analysis

- n: total number of rules, d: total number of fields
- Size of constructed FDD: $O(n^d)$, d is a constant
- For IP packets, d is usually 4
  - Fields: Source IP, Dest. IP, Dest. Port, Protocol Type
- In practice, this worst case is very unlikely to happen because firewall rules are not arbitrary

Summary of Diverse Firewall Design

Step 1: give same requirement to multiple teams to design firewalls

Step 2: compare multiple firewalls to discover all functional discrepancies